Familienname:

Vorname:

Matrikelnummer:

Aufgabe 1 (2 Punkte):
Aufgabe 2 (3 Punkte): Aufgabe 3 (1 Punkt):
Aufgabe 4 (3 Punkte):
Aufgabe 5 (1 Punkt):
Aufgabe 6 (3 Punkte):
Aufgabe 7 (5 Punkte):
Aufgabe 8 (5 Punkte):
Aufgabe 9 (5 Punkte):
Aufgabe 10 (3 Punkte):
Aufgabe 11 (5 Punkte):
Aufgabe 12 (4 Punkte):

Gesamtpunkte (40 Punkte):

Schriftlicher Test (120 Minuten)
VU Einfuhrung ins Programmieren f¨ ur TM¨

28. Juni 2019

Aufgabe 1 (2 Punkte). Was ist eine rekursive Funktion und was darf dabei nicht fehlen? Erlautern¨ Sie das Konzept anhand eines selbstgewahlten Beispiels und geben Sie einen entsprechenden C/C++¨ Code an!

Eine Funktion, die sich selbst aufruft.
Diese benötig eine Abbruchbedingung.

Beispiel: n-te Fibonacci

```
int fib(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n - 1) + fib(n - 2);
}
```

Hinweis. In den folgenden Aufgaben betrachten wir Vektoren $x \in \mathbb{R}^n$ als Objekte der C++ Klasse Vector. Neben Konstruktor, Kopierkonstruktor, Destruktor und Zuweisungsoperator gibt es eine Methode, um die Dimension n auszulesen (size), das Maximum zu bestimmen (max), den Vektor zu partitionieren (partition), den Vektor zu sortieren (sort) und mehrfache Eintrage zu streichen¨ (unique). Die Koeffizienten x_j erhalt man mittels¨ x(j), wobei die Indizes $j = 1,...,n$ im mathematisch ublichen Sinn verwendet werden. Zus¨ atzlich gibt es noch eine Methode (¨ whoAmI), deren Funktionalitat Sie sp¨ ater bestimmen m¨ ussen.¨

```
class Vector {
private : int n;
   double* coeff ;
public :
   Vector (int , double = 0); Vector (const
   Vector&);
   ~Vector ();
   Vector& operator=(const Vector&);

   int size () const ;
   double& operator ()( int ); const double&
   operator ()( int ) const ;

   double max() const ;
   int partition (double );
   void sort (); void
   unique ();
   void whoAmI(double , double );
};
```

Achtung. Der Koeffizientenzugriff mittels Klammer-Operator x(j) wird mit $j = 1,...,n$ indiziert. Der direkte Zugriff auf die Koeffizienten coeff[k] wird (naturlich trotzdem) mit¨ $k = 0,...,n - 1$ indiziert!

Aufgabe 2 (3 Punkte). Schreiben Sie den Konstruktor der Klasse Vector, der einen Vektor $x \in \mathbb{R}^n$ anlegt, wobei der optionale Parameter init der Initialisierungswert fur die Koeffizienten sei (d.h.¨ x_j = init fur alle¨ $j = 1,...,n$). Stellen Sie mittels assert sicher, dass die Dimension $n \geq 0$ ist. Fur¨ $n = 0$ werde der leere Vektor angelegt (d.h. es wird kein Speicher allokiert).

```
Vector::Vector(int n, double x) {
    assert(n >= 0);
    this->n = n;
    this->coeff = new double[n];
    for (int i = 0; i < n; ++i) {
        this->coeff[i] = x;
    }
}
```

Aufgabe 3 (1 Punkt). Schreiben Sie den Destruktor der Klasse Vector.

```
Vector::~Vector() {
    if (this->coeff) {
        delete[] this->coeff;
    }
}
```

Aufgabe 4 (3 Punkte). Schreiben Sie den Zuweisungsoperator der Klasse Vector.

```
Vector& Vector::operator=(const Vector& x) {
    if (this != &x) {
        this->n = x.size();
        if (this->coeff) {
            delete[] this->coeff;
        }
        this->coeff = new double(this->n);
        for (int i = 0; i < this->n; ++i) {
            this->coeff[i] = x.coeff[i];
        }
    }
    return *this;
}
```

Aufgabe 5 (1 Punkt). Schreiben Sie den Zugriffsoperator ( ) fur konstante Objekte der Klasse¨ Vector fur den Koeffizientenzugriff auf¨ x_j mittels x(j). Stellen Sie mittels assert sicher, dass die Koeffizienten im zulassigen Bereich sind, d.h.¨ $j \in \{1,...,n\}$ fur¨ $x \in \mathbb{R}^n$. Beachten Sie, dass der Speichervektor in C++ intern mit $k = 0,...,n-1$ indiziert wird.

```
double& Vector::operator()(int i) {
    assert(i > 0 && i <= this->n);
    return this->coeff[i - 1];
}
```

Aufgabe 6 (3 Punkte). Schreiben Sie die Methode max der Klasse Vector, die das Maximum $\max_{j=1,\dots,n} x_j$ von $x \in \mathbb{R}^n$ zuruckgibt. Stellen Sie mittels¨ assert sicher, dass die Dimension $n > 0$ ist.

```
double Vector::max() const {
    assert(this->n > 0);
    double m = this->coeff[0];
    for (int i = 1; i < this->n; ++i) {
        if (m < this->coeff[i]) {
            m = this->coeff[i];
        }
    }
    return m;
}
```

Aufgabe 7 (5 Punkte). Schreiben Sie die Methode sort der Klasse Vector, die den Vektor $x \in R^n$ aufsteigend sortiert.

Beispiel. x = (1,7,2,5,6,5,9,6) wird durch Aufruf von sort() zu x = (1,2,5,5,6,6,7,9).

```
void Vector::sort()
{
    double temp;
    for (int i = 0; i < this->n; ++i) {
        for (int j = i + 1; j < this->n; ++j) {
            if (this->coeff[i] > this->coeff[j]) {
                temp = this->coeff[i];
                this->coeff[i] = this->coeff[j];
                this->coeff[j] = temp;
            }
        }
    }
}
```

Aufgabe 8 (5 Punkte). Schreiben Sie die Methode unique der Klasse Vector, die einen Vektor $x \in \mathbb{R}^n$ sortiert und alle mehrfach vorkommenden Eintrage¨ x_j streicht. Der Vektor x soll mit dem gekurzten Vektor¨ uberschrieben werden (d.h. Vektorl¨ ange und dynamisches Koeffizientenfeld m¨ ussen¨ ggf. angepasst werden!)

Beispiel. $x = (1,7,2,5,6,5,9,6)$ wird durch Aufruf von unique() zu $x = (1,2,5,6,7,9)$.

```
void Vector::unique()
{
    if (this->n <= 1) return;
    this->sort();
    int k = 1;

    // push unique numbers to the front of coeff array
    for (int i = 1; i < this->n; ++i) {
        if (this->coeff[k-1] != this->coeff[i]) {
            this->coeff[k++] = this->coeff[i];
            cout << *this << endl;
        }
    }

    // create new array with size k
    double* tmp = new double[k];
    for (int i = 0; i < k; ++i) {
        tmp[i] = this->coeff[i];
    }

    // point to new array
    delete[] this->coeff;
    this->n = k;
    this->coeff = tmp;
}
```

Aufgabe 9 (5 Punkte). Schreiben Sie die Methode partition der Klasse Vector. Diese soll zu gegebenem Pivot-Element $y \in \mathbb{R}$ den Vektor $x \in \mathbb{R}^n$ so umordnen, dass alle Elemente die kleiner gleich dem Pivot-Element sind, zuerst im Vektor stehen, und alle Elemente, die großer als das Pivot-Element¨ sind, nachfolgend im Vektor stehen. Zusatzlich soll die Anzahl¨ j der Elemente, die kleiner gleich dem Pivot-Element sind, zuruckgegeben werden. Ihre Methode soll m¨ oglichst geringen Aufwand haben.¨ Vermeiden Sie also, den Vektor zu sortieren!

Beispiel. $x = (1,7,2,5,6,5,9,6)$ wird durch Aufruf von x.partition(5) beispielsweise zu $x = (1,5,2,5,6,7,9,6)$ oder zu $x = (1,2,5,5,7,6,9,6)$, und die Anzahl $j = 4$ wird zuruckgegeben. Be-¨ achten Sie, dass die Anzahl j eindeutig ist, der Ergebnisvektor x aber nicht!

Hinweis. Die Methode partition ist ein wesentlicher Schritt im QuickSort-Algorithmus, den Sie in den Ubungen implementiert haben.¨

```
int Vector::partition(double p)
{
    if (this->n == 0) return 0;
    if (this->n == 1) return (this->coeff[0] <= p ? 1 : 0);

    double tmp;
    int k = 0;
    for (int i = 0; i < this->n; ++i) {
        if (this->coeff[i] <= p) {
            tmp = this->coeff[k];
            this->coeff[k++] = this->coeff[i];
            this->coeff[i] = tmp;
        }
    }

    return k;
}
```

Aufgabe 10 (3 Punkte). Was macht die folgende Methode whoAmI der Klasse Vector?

```
void Vector : : whoAmI(double Cmin, double Cmax) {
    int k = 0; for ( int j =0; j<n; ++ j ) { if ( coeff [ j ] >= Cmin &&
    coeff [ j ] <= Cmax) {
            coeff [ k++ ] = coeff [ j ] ;
       }
    }
    if (k < n) {
       double* tmp = new double [ k ] ;
            for ( int j =0; j<k ;          ++ j ) {
           tmp[ j ] = coeff [ j ] ;
       }
        delete [ ]      coeff ;
       n = k ;
        coeff = tmp;
    }
}
```

Welchen Aufwand hat diese Implementierung der Methode whoAmI für $x \in R^n$? Falls die Funktion für $n = 10^5$ eine Laufzeit von 7 Sekunden hat, welche Laufzeit erwarten Sie für $n = 10^6$?

Die Methode Vector::whoAmI(...) entfernt alle Koeffizienten im Vector, die nicht zwischen Cmin and Cmax liegen. Dh. Cmin <= coeff[j] <= Cmax

Der Aufwand der Methode ist O(n), d.h. linearer Aufwand.

Gegeben ist n = 10^5 mit Laufzeit 5 Sekunden.
Gesucht ist Laufzeit bei n = 10^6 bei liniearem Aufwand.

10^6 / 10^5 = 10 , d.h. Aufwand ist das 10-fache => 7*10 = 70 70 Sekunden

Aufgabe 11 (5 Punkte). Schreiben Sie eine C++ Funktion eratosthenes, die fur gegebenes¨ $n \geq 2$ den Vektor aller Primzahlen $\leq n$ zuruckgibt. Gehen Sie dazu wie folgt vor:¨

- Stellen Sie mittels assert sicher, dass $n \geq 2$ gilt.
- Legen Sie einen Vektor $x = (1,2,3,...,n)$ an.
- Setzen Sie alle Vielfachen von $j = 2,3,4,...$ im Vektor x auf Null.
- Verwenden Sie whoAmI(2,n), um alle Nicht-Primzahlen aus dem Vektor zu eliminieren.

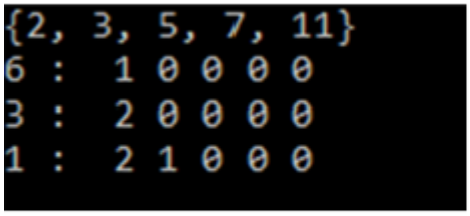
Beispiel. Vector x = eratosthenes(25); soll den Vektor $x = (2,3,5,7,11,13,17,19,23)$ generieren.

```
Vector& eratosthenes(int n) {
    assert(n >= 2);
    Vector* v = new Vector(n);
    for (int i = 0; i < n; ++i) {
        v->setCoeff(i + 1, i + 1);
    }
    for (int i = 2; i < n; ++i) {
        if ((*v)(i) != 0) {
            for (int j = 2; (*v)(i) * j <= n; ++j) {
                (*v).setCoeff((*v)(i) * j, 0);
            }
        }
    }
    (*v).whoAmI(2, n);
    return *v;
}
```

Aufgabe 12 (4 Punkte). Was ist der Shell-Output der folgenden Funktion bei Aufruf Vector y = foo(12)? Was ist die mathematische Funktionalitat der Funktion¨ foo?

```
Vector foo ( int n) {
    Vector x = eratosthenes (n ); Vector y(x . size ());
    for ( int j =1; j<=x . size (); ++ j ) {
        while (n/(( int ) x( j ))*x( j ) == n) {
            n = n/(( int ) x( j )); y( j ) =
            y( j ) + 1; cout << n << " : "
            ;
                for ( int k=1; k<=x . size ();          ++k) {
                cout << " " << y(k );
            }
            cout << endl ;
        }
    }
    return y ;
```

Output:



Die Funktion foo(int n) gibt den Verlauf einer Primzahlzerlegung des Inputs n aus.
Die letzte Zeile zeigt die Anzahl der jeweiligen Primzahlen an.